A Deep-Learning Approach for Parking Slot Detection on Surround-View Images

Conference Paper · June 2019

DOI: 10.1109/IVS.2019.8813777

3 authors, including:

Andrea Zinelli
Università di Parma
4 PUBLICATIONS **14** CITATIONS
SEE PROFILE

Fabio Pizzati
University of Bologna
8 PUBLICATIONS **13** CITATIONS
SEE PROFILE

A Deep-Learning Approach for Parking Slot Detection on Surround-View Images

Andrea Zinelli[1], Luigi Musto[1], Fabio Pizzati[2]

***Abstract*— Being able to automatically detect parking slots during navigation is an important part of an autonomous driving system. Most of the current parking slot detectors either work on images coming from static cameras or are based on hand-designed low-level visual features, limiting the domains of application of such systems considerably. Learning the most suitable features for the task directly from data would allow the system to function in a broader range of situations and be more robust to noise and different observation conditions. This paper presents an end-to-end deep neural network trained to perform automatic parking slot detection and classification on surround-view images, generated by fusing the views of four different cameras positioned on a vehicle. The network architecture is based on the Faster R-CNN baseline. To account for the fact that parking slots can be of different shapes and can be observed at different angles, the predicted bounding boxes are generic quadrilaterals rather than image-aligned rectangles. Moreover, instead of computing offsets wrt anchor boxes, the RPN regresses the region proposals directly. In order to train the network, a small training set comprised of a few hundred images has been manually annotated. The network shows good performance and generalization capability on new examples containing parking slots of the same kinds as those in the training set, suggesting that a wider array of parking slot types could easily be integrated into the system by expanding the dataset.**

I. Introduction

In the last decade, autonomous driving and driver assistant systems have witnessed an enormous increase in popularity and are currently among the most researched technologies. An important requirement for these systems is the ability to detect nearby parking slots automatically, which is becoming an increasingly difficult and critical task as the human population, and therefore the traffic, increases.

One of the most employed solutions commercially is to make use of sonars to detect vacant spaces between obstacles and label those as available parking spaces. While this approach might work well under the supervision of a human operator, it is not reliable in the context of fully automated vehicles, due to the extremely low amount of information returned by this kind of sensor.

Another class of solutions utilizes cameras as sensors and tries to detect available parking slots directly in the image. While the information given by cameras is considerably richer than that of sonars, it comes with the complexity of processing it properly. The detection decision must be based on visual cues, or features, present in the image, and, given the high variability of input data (e.g.: parking slots can be of different shapes, the striping can have different colors, the observation conditions might change, there can be noise *etc*.) hand-designing a set of features that works in a wide array of scenarios is nearly impossible. On the other hand, learning the most appropriate set of features for the task directly from the data would allow for more robustness and adaptability to different conditions. Therefore, this paper presents an end-to-end deep neural network for performing parking slot detection and occupancy classification on surround view images which are generated by fusing the views of four cameras positioned on a vehicle. In particular, since the assumption is that parking slots can have a generic shape and can be observed from any angle, the network is trained to predict the image coordinates of the four corners of each slot. Given the lack of publicly available datasets suited for this kind of task, a small dataset containing a few hundred images was generated and manually annotated.

The paper is structured as follows: Section II reviews some of the work on parking slot detection and classification using computer vision. Section III introduces the approach adopted in this paper, highlighting the current state of the art on deep learning-based object detection techniques on which this work is based. Sections IV and V go into detail on the network architecture chosen as well as on the training logic and the data preparation step. In section VI we discuss the results qualitatively and we highlight the strengths and limits of the current approach.

II. Related Work

A lot of vision-based parking slot detection algorithms work on static images taken, for example, from security cameras[1][2][3]. These systems are typically used to detect vacant slots for parking lot management and are not applicable to dynamic environments, as they rely on strong a priori knowledge: the location of the slots.

Work in the direction of autonomous parking detection during navigation started with [4], in which parking slot markings are segmented based on color. More recent approaches aim at identifying parking slots in surround-view, where a complete representation of the vehicle surroundings is reconstructed by fusing Bird's Eye View (BEV) images coming from multiple cameras. Surround view images allow some geometric properties of parking slots (e.g.: the fact that they appear rectangular, or the fact that the thickness of the markings does not vary due to perspective) to be exploited to obtain more consistent detections. In [5] and [6], for

[1]Andrea Zinelli and Luigi Musto are with University of Parma, Italy {andrea.zinelli1, luigi.musto}@studenti.unipr.it

[2]Fabio Pizzati is with University of Bologna and University of Parma, Italy fabio.pizzati2@unibo.it

example, parking slots are identified using low-level visual features, such as corners or lines. A recently proposed work [7] aims at finding the parking slots by training an AdaBoost [8] classifier to detect the cross-points between parking-line segments, and then proceeds to infer the entry point of each slot from those. In order to train the classifier, a dataset of 8600 images was created, containing the annotations on the position of each cross-point and the orientation of each slot. In [9], the input surround-view image is pre-processed using a Sobel filter in order to identify the pixels potentially belonging to a marking. A probabilistic Hough transform is then applied to extract segments, which are connected to form lines. Finally, the available parking slots are detected using relations between parallel lines.

Most of the cited methods share a common limitation: they are based on hand-designed visual features. As such, they tend to perform poorly outside of the specific or controlled environments for which they are designed. For example, [7] is only able to detect horizontal and vertical slots, so it tends to fail in presence of slanted parking slots or when steering or during the execution of a parking maneuver. [9] tends to perform well in most observation conditions, but it fails to detect slanted slots and requires a computationally intensive post-processing phase. Approaches based on color like [4] might fail in presence of occlusions or noise in the data.

III. Our Approach

In the last few years, deep neural networks have shown outstanding performance in a wide array of computer vision applications, ranging from detection to image generation. One of the most appealing traits of these kinds of algorithms is their ability to learn the most appropriate feature representation directly from the data. As a result, these systems are able to adapt and perform well in a wide variety of environments and under different observation conditions. The fact that parking slots can have different shapes, different kinds of markings and can be observed from different angles makes deep learning particularly suited for parking slot detection.

The parking problem could be framed as a segmentation problem: given an input image, the objective would be to classify each pixel as either belonging to a parking slot or not. However, this type of information is heavily redundant and would require non-trivial post-processing to obtain the final detections.

Seeing as each parking slot can be uniquely identified by its four corners, we propose an end-to-end deep neural network whose task is, given an input image, to output for each parking slot present the coordinates of its four vertices together with a decision on whether the parking slot is free or occupied. In particular, we constrain the inputs to be surround-view images, computed by stitching together four Bird's Eye View (BEV) images coming from four cameras positioned on a vehicle.

The approach is inspired by Faster R-CNN [10], a state of art two-stage object detection deep network upon which a lot of current deep object detectors are based. The first stage

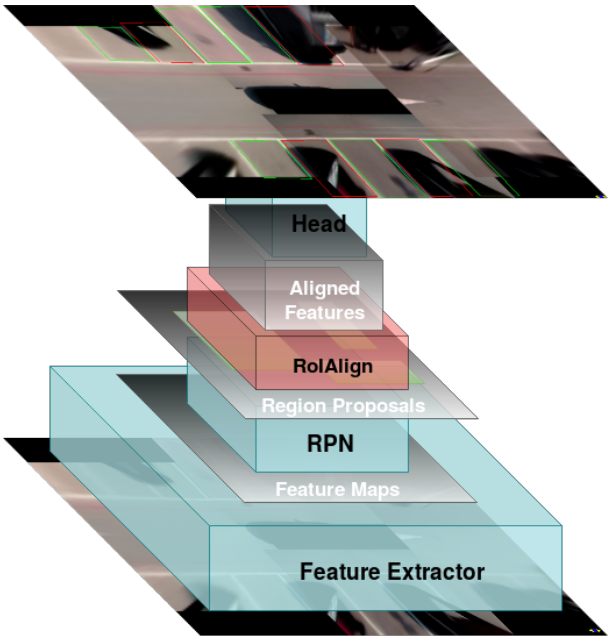


Fig. 1: Schematic representation of the network.

is responsible for generating region proposals, which is a set of bounding boxes that are likely to contain an object of interest. The second stage is responsible for processing each proposal, in order to verify whether it contains an object, classify the object and refine its coordinates. Faster R-CNN is built on anchors, reference boxes used as starting points for determining the region proposals and, ultimately, the final predictions. While an approach based on anchors might work well when the boxes to be predicted are aligned with the image frame, it struggles when they are generic quadrilaterals, as it would require a complex design of anchors of different scales, shapes, and orientations in order to be effective (i.e. [11]). Considering that parking slots can have generic orientations and shapes, inspired by [12] we use an Anchor-Free Region Proposal Network, responsible for regressing the region proposals directly, instead of using anchors as the basis.

Other methods, such as SSD ([13]) and YOLO ([14], [15]) try to predict the bounding boxes directly in a single stage. As a result, they tend to be much faster than two-stage approaches but less accurate and more prone to detecting false positives. Moreover, experiments conducted on a single-stage variant of the proposed network exhibited a considerable degradation of the performance, which is probably due to a combination of limited training set and harder training objective. Considering that our goal is to detect the location of each parking slot as accurately as possible and that real-time is not a hard requirement, we decided against this kind of approaches.

IV. Network Overview

Our proposed deep network is composed of three main modules: a feature extractor backbone, a region proposal

(a) (b) (c) (d) (e)

Fig. 2: Inference pipeline of the network. (a): Visualization of the reference points on the image. (b) Region proposals estimated from the reference points. The green proposals are the ones having objectness score above 0.5. (c): Pruning of the negative region proposals. (d) Remaining positive region proposals after NMS. (e): result of the detection head on the remaining region proposals.

network (RPN), and a detection head. A schematic representation of the network can be seen in Fig. 1.

A. Feature extractor

The feature extractor backbone is responsible for computing a latent feature representation of the input, that will be shared both by the RPN and the detection head. For the backbone network we chose FPN [17]: first, features are extracted using a Resnet34 architecture [16]. Then, a feature pyramid with three levels, $P2$, $P3$ and $P4$ having strides 4, 8 and 16 respectively, is built upon those features, each having 256 channels. Both the RPN and the detection head operate on the $P4$ features. Other approaches, such as [12], build an RPN and a detection head for each level of the pyramid, and each level is responsible for detecting boxes of a specific size. In the case of parking slot detection on surround-view images, we can speculate that the size of the boxes is more or less constant, rendering the multiscale approach redundant.

B. Region Proposal Network

The RPN is responsible for generating good quality region proposals to feed to the detection head in order to obtain the final predictions. More specifically, each pixel in the $P4$ latent space can be mapped to a specific pixel in the input image. These points in the image will be called reference points. Therefore, the RPN job is to determine, for each reference point, whether it belongs to an object (in this case a parking slot) by computing an objectness score between 0 and 1 and, if it does, to regress the four coordinates of the quadrilateral containing the said object. In particular, the RPN is trained to predict the normalized offsets of the four vertices of the quadrilateral wrt each reference point. For instance, let $r = (r_x, r_y)$ be the image coordinates of a specific reference point, and let $G = \{g_i = (g_{x,i}, g_{y,i}), i \in \{1, 2, 3, 4\}\}$ be the image coordinates of the four vertices of the box to be predicted. The RPN is trained to predict the following values:

$$p = \left\{ p_i = \frac{g_i - r}{\gamma}, i \in \{1, 2, 3, 4\} \right\} \tag{1}$$

where γ is a normalization constant. Empirically, we chose $\gamma = 50$. This logic has been implemented as two convolutional branches, a classification branch and a regression branch, each one comprised of two convolutional layers. In both branches, the first layer is represented by a 3 x 3 kernel having 256 output channels, padding 1 and followed by batch normalization [19] and ReLU. The second layer is a 1 x 1 kernel having 1 output channel for the classification branch, and 8 output channels for the regression branch. The output of both branches has the same spatial size as the $P4$ features, allowing to naturally represent the classification and regression decisions for each reference point in the image.

C. Detection Head

The detection head is responsible for processing high-quality proposals generated by the RPN and refine them. Firstly, each negatively labeled region proposal (having a classification score below 0.5) is removed. Secondly, Non-Maximum Suppression with an Intersection over Union (IoU) threshold of 0.5 is applied to the remaining proposals in order to remove duplicate detections. For simplicity and efficiency, the NMS computations are done considering the circumscribed rectangles instead of the original quadrilaterals. Thirdly, RoIAlign (see [18]) is used in order to extract 7 x 7 features from $P4$ for each remaining proposal. Again, RoIAlign is applied to the circumscribed rectangles. Finally, each extracted feature map is processed by the detection head, which classifies the object (in this case it decides whether the object is background, an occupied slot or a free slot) and regresses the final coordinates for the object. Like the RPN, the detection head is also trained to estimate normalized offsets. The only difference is that, in this case, the offsets are computed wrt the centroids of the corresponding region proposals instead of wrt to the reference point positions. The normalization constant is the same used for the RPN. This portion of the network has been implemented as two fully connected layers, each having 2048 neurons, batch normalization and ReLU, followed by two sibling fully-connected layers responsible for the classification and regression tasks, having 3 and 8 outputs respectively.

V. Training

A. Data preparation

In order to train the network, a small dataset composed of around 400 images, containing different kinds of slots observed from various points of view, has been manually annotated with the coordinates of the corners of each parking slot, as well as the information on the occupancy of the slots.

The target box coordinates have been preprocessed in order to have a consistent ordering among different examples. More specifically, the strategy adopted for the ordering consists in sorting the points in clockwise order starting from the top-left point, which is determined by its angle wrt the centroid of the box.

Ensuring that the ordering among examples is consistent is necessary to make sure there is no ambiguity during training. Unlike image-aligned boxes, which can be represented by their center, height and width, in generic quadrilaterals it is mandatory for the network to be able to determine in which order to predict the points and which point is first.

B. Training logic

During the training phase of the network, not all reference points and regions proposals are used to update the weights of the network.

For each training iteration, given a batch of images, the loss for the RPN is computed on 256 reference points sampled among all the reference points in all images in the batch such that the ratio between positive and negative examples (according to ground truth) is 1:1. In case that less than 128 positive reference points are present in a batch, enough negative reference points are sampled to reach 256 elements total. The loss function for each reference point is defined as follows:

$$L(c, c^*, p, p^*) = L_{cls}(c, c^*) + \lambda[c^* = 1]L_{reg}(p, p^*) \quad (2)$$

where c and c^* are the predicted and ground truth objectness probabilities for each sampled reference point, and p and p^* are the predicted and ground truth normalized coordinates (see Eq. 1) of the corrisponding boxes. The classification loss L_{cls} used is a binary cross-entropy loss, defined as:

$$L_{cls}(c, c^*) = -c^* \cdot \log c - (1 - c^*) \cdot \log(1 - c). \quad (3)$$

The regression loss L_{reg} is a smooth-L1 loss, defined as:

$$L_{reg}(p, p^*) = \frac{1}{4} \sum_{i=1}^{4} \text{Smooth}_{L1}(p_i - p_i^*), \quad (4)$$

where

$$\text{Smooth}_{L1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise.} \end{cases} \quad (5)$$

The Iverson bracket function $[c^* = 1]$ evaluates to 1 for all the reference points contained in a ground truth box, and 0 otherwise. This has the effect of ignoring the regression loss for negative reference points, as they have no corresponding bounding box. λ is a balancing hyperparameter which we empirically set to $\lambda = 3$. The total loss for the RPN is given by the normalized sum of the contributions of all sampled reference points.

Likewise, the detection head is trained using a sampling strategy. First, all the region proposals generated by the RPN are classified into positive and negative samples according to their overlap with the ground truth boxes. In particular, for each proposal, the ground truth box having maximum IoU with it is found. Then, if the IoU value is above 0.5, the proposal is considered positive and is associated with that box. Otherwise, it is considered a negative example. As with the NMS, the IoU values are computed using the corresponding circumscribed rectangles to each quadrilateral. A balanced batch of 128 region proposals between positive and negative examples is then sampled for training the detection head. Again, should not enough positive proposals be present, the batch is padded with negative samples. The features corresponding to each selected proposal box are then extracted using RoIAlign and fed to the detection head in order to compute the loss. The loss function for the detection head is the same one used for the RPN, except for the fact that L_{cls} is now a multi-class cross-entropy, defined as:

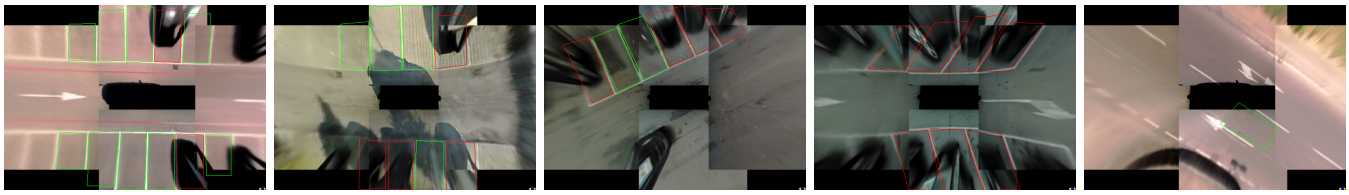
$$L_{cls}(C) = -\log \cdot c_t \quad (6)$$

where C is the predicted probability vector and c_t is the element of C corresponding to the true class. The balancing parameters are the same. The total loss for the detection head is, again, given by the normalized sum of the losses of all sampled proposal boxes. The total loss used for training the network is the sum of the RPN loss and the detection head loss.

The network was jointly trained from scratch using the standard Stochastic Gradient Descent (SGD) algorithm with momentum 0.9, batch size 16 and learning rate 10^{-3} for 100K iterations. Despite this number of iterations may seem excessive for such a small dataset, the sampling procedure adopted during the optimization procedure ensures that there is enough stochasticity during training to make it worthwhile. Data augmentation was employed. In particular, horizontal and vertical flips are applied independently, each with probability 0.5. Furthermore, brightness, saturation, and contrast are perturbed independently in the range of $\pm 20\%$. The spatial dimension of the input images is 384 x 544.

VI. Empirical results

To demonstrate the capability of our method to function under different observation conditions, in presence of noise and for different kinds of parking slots, we tested the network on unobserved parking spaces of various kinds. Some of the results are shown in Fig. 3. Despite the very limited dataset used for training, the network exhibits a remarkable ability to generalize to new environments containing parking slots that are similar to those observed during training. In particular, the network is currently able to handle parking slots displaying different patterns (Fig. (3b)) and rotated slots (Fig. (3c)). Moreover, despite the fact that the training set contains only 20 examples of slanted parking slots, the network is able to detect them with acceptable accuracy, as shown in Fig. (3d). The system is also able to withstand noise: in Figs. (3b) to (3d) the cameras are not properly calibrated, which results in misalignments between the images, and some dirt is present on the lenses. Fig. (3e) shows a failure case in which road markings are erroneously interpreted as a free parking slot. Moreover, there is a specific set of parking slot orientations for which the network exhibits



(a) (b) (c) (d) (e)

Fig. 3: Results of the network on different types of parking slots under different observation conditions: (a) the most common scenario. (b) Different pattern. (c) Rotation. (d) Slanted parking slots. (e) Failure case.

erratic behavior. This phenomenon is shown in Fig. 4 and is probably induced by the ordering imposed to the predicted points during training: close to some critical observation angles the network is not able to identify which point to predict first, hence it tends to estimate something in between the correct coordinates.

The network is able to run on an NVIDIA Geforce GTX 1080 at over 13 FPS.

A. Ablation study

We performed an ablation study to determine how important the detection head and the sampling strategy adopted are to the overall accuracy of the network. In particular, we carried out two distinct experiments:

- first, we gave the entire responsibility for the prediction to the RPN by removing the detection head and by training the classification branch of the RPN to predict the class directly, instead of just outputting an objectness score. This network was trained for the same amount of iterations as the original one, following the same procedure with the difference that now the classification loss for the RPN is multi-class cross entropy;
- second, we trained the network without the detection head, this time using all the reference points at each iteration, instead of sampling a balanced set of positive and negative examples.

Point sampling	Second stage	mAP	mAP_{50}	mAP_{70}
✓	✓	**44.9**	**60.2**	**54.8**
✓		20.1	36.5	25.4
		14.1	29.6	17.4

TABLE I: Results over the test set. mAP_{50} and mAP_{70} represent the mean Average Precision using IoU threshold of 50% and 70%, respectively. mAP represents the mean Average Precision averaged over multiple thresholds (from 50% to 95%, in intervals of 5%).

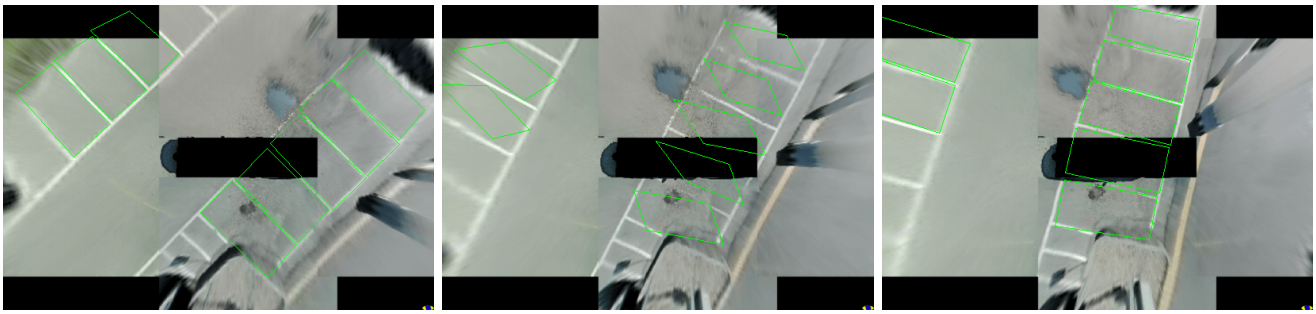
To evaluate the performance of each experiment, a test set of 107 images of unobserved parking spaces was annotated. The results of this study, displayed in Table I, show a considerable degradation of the detection quality wrt the full pipeline, which becomes especially substantial in the second experiment. An example of the different behaviors of the network in the three cases can be observed in Fig. 5. The loss of precision in the first case can be explained by the fact that the final predictions are no longer based on the ad-hoc crops extracted by RoIAlign, but rather on generic patches of the feature maps. This limits both the richness and localization accuracy of the features used for detection, resulting in a worse detection quality and more false positives. The further decrease in performance in the second experiment is due to the loss of the beneficial effects of the sampling strategy on training: on one hand, it balances the positive and negative examples, aiding convergence. On the other hand, it provides stochasticity, favoring generalization especially when the training set is limited. Furthermore, the relative loss in mAP when increasing the IoU threshold is much higher in the versions without the second stage, which implies an overall significantly better detection accuracy by the full network. This suggests that single-stage approaches, such as [13] and [14], might be weaker than two-stage pipelines for this kind of problem with limited data. The blurring at the edges of the images, caused by the BEV transformation, may cause some detection instability in those regions, resulting in mAP scores that are not representative of the true performance of the full network. Moreover, some false positives might still be detected due to the network being exposed to unobserved road markings that look similar to parking slots (e.g. Fig. 3e). However, these false detections usually last no more than a couple of frames, and can therefore be filtered out easily. Increasing the size of the training set and training the network on a higher variety of road markings should drastically reduce the number of false positives.

It was further noticed that the presence of the detection head has a beneficial effect on the RPN during training: the RPN tends to converge faster and have better standalone performance when trained together with the detection head.

VII. Conclusions

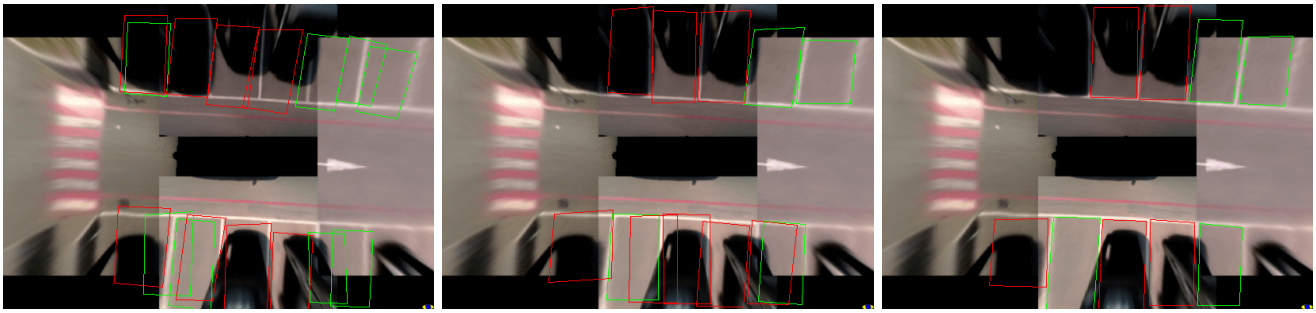
In this paper we presented an end-to-end deep learning-based approach for parking slot detection and occupancy classification on surround view images. The network employed is an anchor-free variant of the Faster R-CNN framework, adapted to handle generic quadrilaterals instead of image-aligned bounding boxes. The network has been trained on a small dataset comprised of around 400 surround-view images, generated by stitching together four Bird's Eye View images coming from four cameras placed on a vehicle.

The main advantage of this method over the current state of the art for parking slot detection is its ability to generalize to, and accurately detect, different kinds of slots, like



(a) (b) (c)

Fig. 4: Erratic behavior of the network at some specific observation angles. The prediction is correct for most observation angles (Figs. (a) and (c)), but fails when approaching some specific critical orientations (Fig. (b)).



(a) (b) (c)

Fig. 5: Results of the ablation study. (a): Network without the detection head using all reference points. (b): Network without the detection head using the usual sampling strategy for the RPN. (c): Full pipeline.

parallel, perpendicular and slanted, under different lighting and observation conditions.

The system exhibited a remarkable capability to adapt to new scenarios containing parking slots of the same type as those contained in the training set as well as robustness to noise and misalignments between the stitched images. This suggests that expanding the training data could allow the network to recognize the vast majority of parking slot types.

An interesting research direction would be to try to detect the parking slots directly on the original images, instead of their BEV.

REFERENCES

[1] N. True, Vacant parking space detection in static images, 2017.

[2] M. Tschentscher and M. Neuhausen, Video-based Parking-space Detection, 2012.

[3] K. Yamada and M. Mizuno, A vehicle parking detection method using image segmentation, Electronics and Communications in Japan (Part III: Fundamental Electronic Science), Vol. 84, pp. 25-34, October 2001.

[4] X. Jin, C. Guang and X. Ming, Vision-guided automatic parking for smart car, Proceedings of the IEEE Intelligent Vehicles Symposium 2000 (Cat. No.00TH8511), pp. 725-730, October 2000.

[5] G. Jung, H. and S. Kim, D. and J. Yoon, P. and K. Jaihie, Structure Analysis Based Parking Slot Marking Recognition for Semi-automatic Parking System, August 2006.

[6] R. Yusnita, F. Norbaya and N. Basharuddin, Intelligent Parking Space Detection System Based on Image Processing, International Journal of Innovation, Management and Technology, pp. 232-235, 2012.

[7] L. Li, L. Zhang, X. Li, X. Liu, Y. Shen and L. Xiong, Vision-based parking-slot detection: A benchmark and a learning-based approach, 2017 IEEE International Conference on Multimedia and Expo (ICME), pp. 649-654, July 2017.

[8] Y. Freund and R. E Schapire, A Short Introduction to Boosting, Journal of Japanese Society for Artificial Intelligence, Vol. 14, pp. 771-780, October 1999.

[9] K. Hamada, Z. Hu, M. Fan and H. Chen, Surround view based parking lot detection and tracking, 2015 IEEE Intelligent Vehicles Symposium (IV), pp. 1106-1111, Seoul, 2015.

[10] S. Ren., K. He, R. Girshick and J. Sun, Faster R-CNN: Towards Real-time Object Detection with Region Proposal Networks, Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1, Vol. 84, pp. 91-99, 2015.

[11] Y. Liu and L. Jin, Deep Matching Prior Network: Toward Tighter Multi-oriented Text Detection, July 2017.

[12] Z. Zhong and L. Sun, An Anchor-Free Region Proposal Network for Faster R-CNN based Text Detection Approaches, 2018.

[13] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Fu and A. C. Berg, SSD: Single Shot MultiBox Detector, Computer Vision - ECCV 2016, pp. 21-37, 2016.

[14] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, You Only Look Once: Unified, Real-Time Object Detection, CVPR, June 2016.

[15] J. Redmon, A. Farhadi, YOLO9000: Better, Faster, Stronger, December 2016.

[16] K. He, X. Zhang, S. Ren and J. Sun, Deep Residual Learning for Image Recognition, 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770-778, 2016.

[17] T. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan and S. Belongie, Feature Pyramid Networks for Object Detection, 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 936-944, 2017.

[18] K. He, G. Gkioxari, P. Dollár and R. Girshick, Mask R-CNN, 2017 IEEE International Conference on Computer Vision (ICCV), pp. 2980-2988, 2017.

[19] S. Ioffe and C. Szegedy, Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, February 2015.